

XI`AN TECHNOLOGICAL UNIVERSITY

课程设计报告

课程名称：

专 业：

班 级：

姓 名：

学 号：

指导教师： 刘敏侠

成 绩：

2025 年 6 月 20 日

1. 问题描述

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | C语言项目实践 | | | | | 指导教师 | | 刘敏侠 | |
| 课设题目1 | 学生信息管理系统 | | | | | | | |  |
| 学号 |  | 姓名 |  | 地点 |  | | 完成时间 | | 2025/6/16  2025/6/20 |
| 设计任务 | 建立一个学生信息管理系统，利用单链表实现，具有如下的五大功能模块，具体要求如下：  qq_pic_merged_1687695704408  学生信息管理系统 | | | | | | | | |
| 设计要求 | 1. 输入记录模块   逐个输入学生基本信息，学生信息包含：学号、姓名、性别、年龄、班级等。   1. 查询记录模块   按照学生的学号或者姓名进行查询，把满足条件的学生信息显示出来。   1. 更新记录模块   修改某一个学生的信息，插入一个学生信息；删除一个学生的信息。   1. 统计记录模块   能够按班级、性别统计学生数量，并输出统计结果；。   1. 输出记录模块   显示所有学生的信息。   1. 能够将所有学生信息保存到文件中，并能从文件中读取学生信息；   以上为最基本要求，可根据自己的理解在此基础上对各个模块增加和完善相应的功能，使得整个系统能更加接近实际的应用。 | | | | | | | | |
| 成果要求 | 1. 设计出完整流程图。  2．编写出全部模块对应的函数。  3．在集成开发工具中调试通过。  4．参加答辩。  5．写出完整、规范的课程设计报告，在指定的时间内提交指导教师。 | | | | | | | | |
| 设计报告  要求 | 1．封面：（格式附后）  2．课程设计任务书  3．课程设计报告：  ①题目用黑体三号，段后距18磅（或1行），居中对齐；  ②标题用黑体四号，段前、段后距6磅（或0.3行）；  ③正文用小四号宋体，行距为1.25倍行距；  ④标题按“一”、“㈠”、“1”、“⑴”顺序编号。  ⑤报告和软件打包以学号姓名命名，邮件主题也为学号姓名。  ⑴ 系统总体方案  ①题目用黑体三号，段后距18磅（或1行），居中对齐；  ②标题用黑体四号，段前、段后距6磅（或0.3行）；  ③正文用小四号宋体，行距为1.25倍行距；  ④标题按“一”、“㈠”、“1”、“⑴”顺序编号。  ⑤报告和软件打包以学号姓名命名，邮件主题也为学号姓名。  ①题目用黑体三号，段后距18磅（或1行），居中对齐；  ②标题用黑体四号，段前、段后距6磅（或0.3行）；  ③正文用小四号宋体，行距为1.25倍行距；  ④标题按“一”、“㈠”、“1”、“⑴”顺序编号。  ⑤报告和软件打包以学号姓名命名，邮件主题也为学号姓名。  ⑵ 设计思路和主要步骤  ⑶ 各功能模块和流程图  ⑷ 设计代码  ⑸ 心得体会 | | | | | | | | |

1. 算法设计

**1.设计思路：**

整体思路是“内存操作 + 文件持久化”。用户的所有增、删、改、查操作都在内存中的链表上进行，以保证高效的响应速度。当用户需要保存工作或程序退出时，通过文件操作模块将内存中的数据写入磁盘文件。程序下次启动时，首先执行加载操作，将磁盘文件中的数据恢复到内存的链表中。

**2.主要步骤：**

1. **需求分析**：明确系统功能，特别是文件存取的需求。
2. **数据结构定义**：在 student.h 中定义 Student 和 Node 结构体。
3. **底层函数实现**：在 student.c 中实现链表的基本操作（增、删、遍历等）。
4. **文件操作实现（新增重点）**：
   1. **实现** saveToFile **函数**：使用 fopen 以写入模式("w")打开文件。遍历链表，使用 fprintf 将每个节点的数据格式化地写入文件。
   2. **实现** loadFromFile **函数**：使用 fopen 以读取模式("r")打开文件。循环使用 fscanf 逐条读取文件中的记录，每读取一条就 malloc 一个新节点，填充数据后插入到链表尾部。需要处理文件不存在的初始情况。
5. **上层功能模块开发**：基于链表基本操作，构建输入、查询、更新、统计等功能。
6. **主函数与菜单集成**：在 main.c 中，在程序开始时**立即调用** loadFromFile **函数**。在主菜单中添加“保存信息到文件”的选项，并将其链接到 saveToFile 函数。
7. **整合与调试**：编译所有文件，并进行全面测试。重点测试：
   1. 程序首次运行（无 student.txt 文件）是否正常。
   2. 添加几条数据后，保存是否成功。
   3. 关闭程序再重新打开，数据是否被成功加载。
   4. 修改或删除数据后，再次保存，文件内容是否被正确覆盖。

#### ****3.功能模块算法设计****

根据课程设计任务书的要求 ，对各核心功能模块的算法设计如下：

**4.输入记录模块 (**addStudent**)**

* **目标**：向链表中添加一个新的学生记录。
* **算法**：
  1. 使用 malloc 动态分配一个新的 Node 节点内存。
  2. 提示用户输入学号、姓名等信息，并存入新节点的数据域。
  3. 唯一性检查：为保证数据完整性，遍历整个链表，检查新输入的学号是否已经存在。如果存在，则提示用户错误，并 free 之前分配的节点内存，终止添加 。
  4. **插入节点**：如果学号唯一，则从头节点开始遍历链表，找到最后一个节点（其 next 指针为 NULL）。
  5. 将最后一个节点的 next 指针指向新创建的节点，完成尾部插入。

**5.更新记录模块 (**modifyStudent**,** deleteStudent**)**

* **目标**：修改或删除指定的学生记录。
* **修改算法 (**modifyStudent**)**：
  1. 提示用户输入要修改的学生的学号。
  2. 遍历链表，使用 strcmp 函数比较学号，查找目标节点。
  3. 若找到，则依次提示用户输入新的姓名、性别、年龄等信息，直接更新该节点数据域中的内容 。
  4. 若未找到，则向用户返回提示信息。
* **删除算法 (**deleteStudent**)**：
  1. 提示用户输入要删除的学生的学号。
  2. 定义两个指针：前驱指针 prev 和当前指针 curr。prev 初始指向头节点，curr 指向第一个数据节点。
  3. 同时移动 prev 和 curr 指针遍历链表，查找学号匹配的 curr 节点。
  4. 找到后，执行删除操作：将前驱节点的 next 指向当前节点的 next (prev->next = curr->next;)，从而将 curr 节点从链中断开。
  5. 调用 free(curr); 释放被删除节点的内存，防止内存泄漏 。
  6. 若未找到，则向用户返回提示信息。

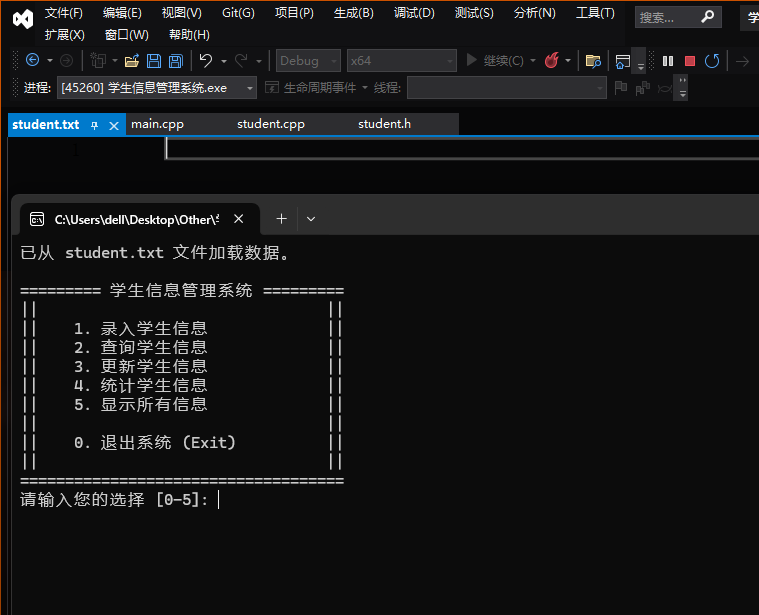
**6.查询与统计模块**

* **查询算法 (**searchStudentById**,** searchStudentByName**)**：
  1. 提示用户输入查询关键字（学号或姓名）。
  2. 从第一个数据节点开始遍历链表。
  3. 在循环中，使用 strcmp 比较关键字与当前节点的数据。
  4. 如果匹配，则打印该节点的全部信息。对于姓名查询，不立即终止，而是继续遍历以找出所有同名学生 。
* **统计算法 (**statisticsByClass**,** statisticsByGender**)**：
  1. 初始化一个计数器 count 为0。
  2. 遍历整个链表。
  3. 对于每一个节点，判断其信息（班级或性别）是否符合统计条件。
  4. 如果符合，则将 count 加一。
  5. 遍历结束后，输出最终的统计结果 。

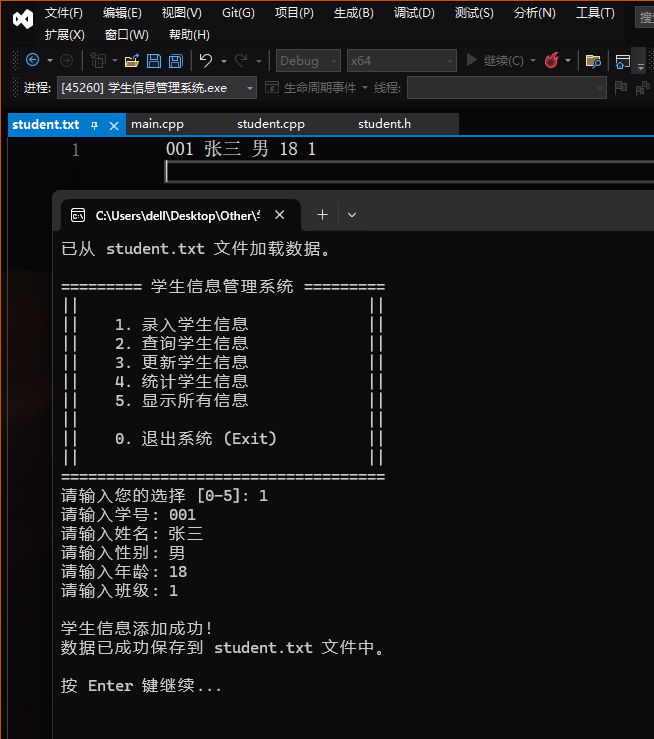
**7.文件持久化模块 (**saveToFile**,** loadFromFile**)**

* **保存算法 (**saveToFile**)**：
  1. 以“写入”模式 ("w") 打开指定文件（如student.txt）。此模式会清空文件原有内容。
  2. 遍历链表（从第一个数据节点开始）。
  3. 对每个节点，使用 fprintf 函数将学生的各项信息按预定格式（如以空格分隔）写入文件，并在末尾添加换行符 。
  4. 遍历结束后，关闭文件。
* **加载算法 (**loadFromFile**)**：
  1. 在程序启动时调用此函数。
  2. 以“读取”模式 ("r") 打开指定文件。
  3. 若文件打开失败（例如首次运行，文件不存在），则打印提示信息并返回一个空的头节点 。
  4. 若文件打开成功，则进入一个 while 循环，条件为 fscanf 函数成功从文件中读取到一条完整的学生记录。
  5. 循环体内，为每条读取的记录动态分配一个新的 Node 节点，并将数据存入其中。
  6. 将新节点插入到链表的尾部。
  7. 循环结束后，关闭文件，返回构建好的带有数据的链表 。

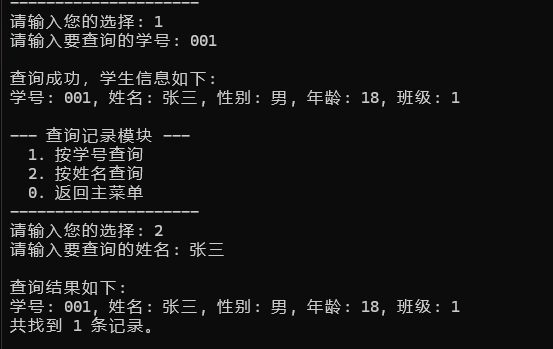
1. 算法实现
2. 程序主页面，启动与加载：程序启动后，首先显示从 student.txt 文件加载数据的提示，然后展示主菜单，等待用户操作 。



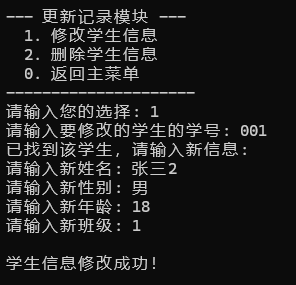
1. 录入学生信息，信息录入：下图展示了录入一个学号为"001"的学生信息的完整过程。添加成功后，系统提示数据已保存到文件 。

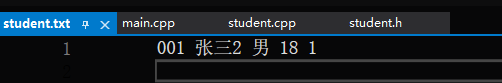


1. 查询学生信息，信息查询：系统提供了按学号和姓名两种查询方式。下图演示了分别使用这两种方式查询名为“张三”的学生，并准确返回了结果 。

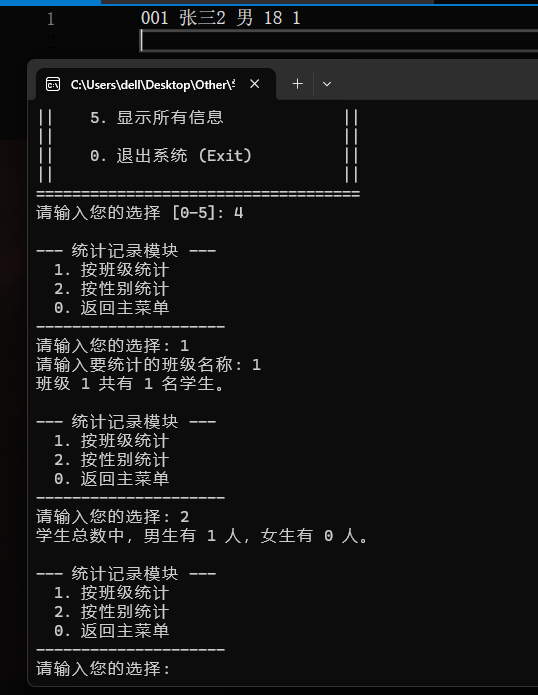


1. 更新学生信息，信息更新：用户可以修改或删除学生信息。下图展示了将学号"001"的学生的姓名从“张三”修改为“张三2”的操作。修改成功后，student.txt 文件中的内容也同步更新





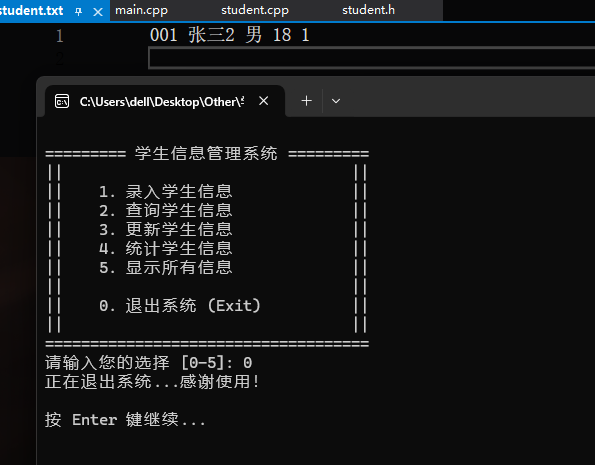
1. 统计学生信息，信息统计与显示：系统能够按班级和性别进行统计，也可以完整地列出所有学生的信息，方便用户全局浏览 。

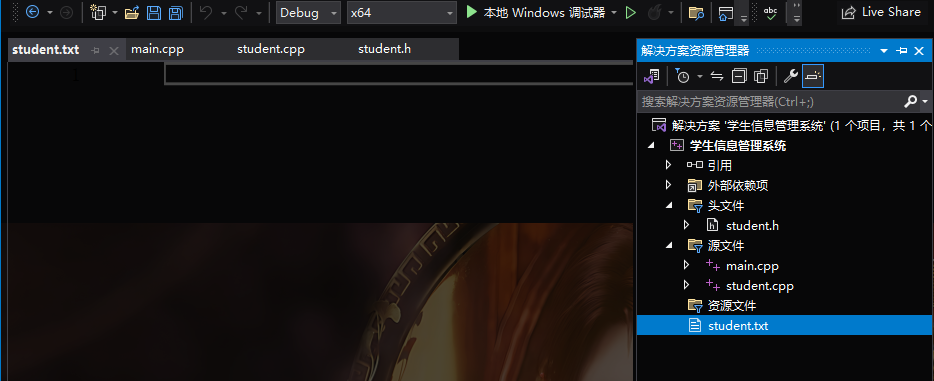


1. 显示所有信息



1. 退出系统 Exit





四．心得体会

通过本次C语言课程设计，我不仅加深了对C语言基础和链表数据结构的理解，更重要的是，我成功地为系统添加了文件操作功能，实现了数据的持久化存储。这个过程让我收获颇丰。

首先，在实现文件读写功能时，我深刻体会到了数据格式统一的重要性。在 saveToFile 函数中，我使用 fprintf 以“字符串 空格 字符串 空格 ...”的格式写入数据，那么在 loadFromFile 函数中，就必须使用与之完全匹配的 fscanf 格式字符串来读取，任何一个字段的顺序或类型不匹配，都会导致数据读取混乱甚至程序崩溃。这让我明白了接口和协议在编程中的重要性。

其次，我学会了处理文件操作中的各种边界情况。比如，程序第一次运行时，数据文件 students.dat 是不存在的，fopen 会返回 NULL。我必须在代码中正确处理这种情况，不是报错退出，而是将其视为一个正常的初始状态，返回一个空链表让系统继续运行。这培养了我编写健壮（Robust）程序的意识。

此外，将加载功能放在程序启动时自动执行，将保存功能作为菜单选项供用户手动触发，是一种经典且用户友好的设计模式。我还额外添加了在退出时提醒用户保存的功能，这让程序的人性化体验又上了一个台阶。

总而言之，这次对文件操作的实践，让我真正将C语言的知识应用到了一个“看得见、摸得着”的实际问题上。它不再是冷冰冰的链表节点，而是可以被保存、被恢复的学生记录。这次经历极大地提升了我的自信心，也让我对软件开发的完整流程——从内存操作到磁盘存储——有了更全面的认识。

五．参考文献

[1]张学孟,梅诗冬,吕家威.基于在线信息化平台的学生信息管理系统运用研究[J].信息系统工程,2025,(06):8-11.

[1]王英.C语言中循环转递归函数策略研究[J].科学技术创新,2025,(15):71-74.

[1]单宝华,佟智慧.C语言在线编程代码缺陷检测方法研究[J].网络安全技术与应用,2025,(05):56-59.

[1]夏黎晨.编程语言中的深拷贝和浅拷贝对算法设计的影响研究[J].软件,2025,46(04):151-153.

[1]葛方振,洪留荣.C语言程序设计基础实验教程[M].中国铁道出版社:202212.108.

源代码

1. student.h (头文件 / 链接口)
2. #ifndef STUDENT\_H
3. #define STUDENT\_H
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <string.h>
7. *// 学生信息结构体*
8. typedef struct Student {
9. char id[20];
10. char name[50];
11. char gender[10];
12. int age;
13. char className[50];
14. } Student;
15. *// 链表节点结构体*
16. typedef struct Node {
17. Student data;
18. struct Node\* next;
19. } Node;
20. *// === 函数原型声明 (模块接口) ===*
21. *// 1. 输入模块*
22. void addStudent(Node\* head);
23. *// 2. 查询模块*
24. void queryMenu(Node\* head);
25. *// 3. 更新模块*
26. void updateMenu(Node\* head);
27. *// 4. 统计模块*
28. void statisticsMenu(Node\* head);
29. *// 5. 输出模块*
30. void displayAllStudents(Node\* head);
31. *// 6. 文件操作模块 (数据持久化)*
32. void saveToFile(Node\* head, const char\* filename); *// 保存数据到文件*
33. Node\* loadFromFile(const char\* filename);         *// 从文件加载数据*
34. *// 辅助函数*
35. void clearInputBuffer();
36. void freeList(Node\* head);
37. #endif *// STUDENT\_H*
38. student.cpp (函数实现文件)
39. #define \_CRT\_SECURE\_NO\_WARNINGS
40. #include "student.h"
41. *// 清空输入缓冲区*
42. void clearInputBuffer() {
43. int c;
44. while ((c = getchar()) != '\n' && c != EOF);
45. }
46. *// 检查学号是否唯一*
47. int isIdUnique(Node\* head, const char\* id) {
48. Node\* p = head->next;
49. while (p != NULL) {
50. if (strcmp(p->data.id, id) == 0) {
51. return 0; *// 不唯一*
52. }
53. p = p->next;
54. }
55. return 1; *// 唯一*
56. }
57. *// 1. 输入记录模块*
58. void addStudent(Node\* head) {
59. Node\* newNode = (Node\*)malloc(sizeof(Node));
60. if (newNode == NULL) {
61. printf("内存分配失败！\n");
62. return;
63. }
64. printf("请输入学号: ");
65. scanf("%s", newNode->data.id);
66. clearInputBuffer();
67. if (!isIdUnique(head, newNode->data.id)) {
68. printf("错误：该学号已存在！\n");
69. free(newNode);
70. return;
71. }
72. printf("请输入姓名: ");
73. scanf("%s", newNode->data.name);
74. clearInputBuffer();
75. printf("请输入性别: ");
76. scanf("%s", newNode->data.gender);
77. clearInputBuffer();
78. printf("请输入年龄: ");
79. scanf("%d", &newNode->data.age);
80. clearInputBuffer();
81. printf("请输入班级: ");
82. scanf("%s", newNode->data.className);
83. clearInputBuffer();
84. newNode->next = NULL;
85. *// 找到链表尾部并插入*
86. Node\* p = head;
87. while (p->next != NULL) {
88. p = p->next;
89. }
90. p->next = newNode;
91. printf("\n学生信息添加成功！\n");
92. }
93. *// 2. 查询记录模块*
94. void searchStudentById(Node\* head) {
95. if (head->next == NULL) {
96. printf("当前无任何学生信息。\n");
97. return;
98. }
99. char id[20];
100. printf("请输入要查询的学号: ");
101. scanf("%s", id);
102. clearInputBuffer();
103. Node\* p = head->next;
104. int found = 0;
105. while (p != NULL) {
106. if (strcmp(p->data.id, id) == 0) {
107. printf("\n查询成功，学生信息如下:\n");
108. printf("学号: %s, 姓名: %s, 性别: %s, 年龄: %d, 班级: %s\n",
109. p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);
110. found = 1;
111. break;
112. }
113. p = p->next;
114. }
115. if (!found) {
116. printf("未找到学号为 %s 的学生。\n", id);
117. }
118. }
119. void searchStudentByName(Node\* head) {
120. if (head->next == NULL) {
121. printf("当前无任何学生信息。\n");
122. return;
123. }
124. char name[50];
125. printf("请输入要查询的姓名: ");
126. scanf("%s", name);
127. clearInputBuffer();
128. Node\* p = head->next;
129. int found = 0;
130. printf("\n查询结果如下:\n");
131. while (p != NULL) {
132. if (strcmp(p->data.name, name) == 0) {
133. printf("学号: %s, 姓名: %s, 性别: %s, 年龄: %d, 班级: %s\n",
134. p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);
135. found++;
136. }
137. p = p->next;
138. }
139. if (found == 0) {
140. printf("未找到姓名为 %s 的学生。\n", name);
141. }
142. else {
143. printf("共找到 %d 条记录。\n", found);
144. }
145. }
146. void queryMenu(Node\* head) {
147. int choice;
148. do {
149. printf("\n--- 查询记录模块 ---\n");
150. printf("  1. 按学号查询\n");
151. printf("  2. 按姓名查询\n");
152. printf("  0. 返回主菜单\n");
153. printf("---------------------\n");
154. printf("请输入您的选择: ");
155. scanf("%d", &choice);
156. clearInputBuffer();
157. switch (choice) {
158. case 1: searchStudentById(head); break;
159. case 2: searchStudentByName(head); break;
160. case 0: break;
161. default: printf("无效选择，请重新输入。\n");
162. }
163. } while (choice != 0);
164. }
165. *// 3. 更新记录模块*
166. void modifyStudent(Node\* head) {
167. if (head->next == NULL) {
168. printf("当前无任何学生信息。\n");
169. return;
170. }
171. char id[20];
172. printf("请输入要修改的学生的学号: ");
173. scanf("%s", id);
174. clearInputBuffer();
175. Node\* p = head->next;
176. int found = 0;
177. while (p != NULL) {
178. if (strcmp(p->data.id, id) == 0) {
179. printf("已找到该学生，请输入新信息:\n");
180. printf("请输入新姓名: ");
181. scanf("%s", p->data.name);
182. clearInputBuffer();
183. printf("请输入新性别: ");
184. scanf("%s", p->data.gender);
185. clearInputBuffer();
186. printf("请输入新年龄: ");
187. scanf("%d", &p->data.age);
188. clearInputBuffer();
189. printf("请输入新班级: ");
190. scanf("%s", p->data.className);
191. clearInputBuffer();
192. printf("\n学生信息修改成功！\n");
193. found = 1;
194. break;
195. }
196. p = p->next;
197. }
198. if (!found) {
199. printf("未找到学号为 %s 的学生。\n", id);
200. }
201. }
202. void deleteStudent(Node\* head) {
203. if (head->next == NULL) {
204. printf("当前无任何学生信息。\n");
205. return;
206. }
207. char id[20];
208. printf("请输入要删除的学生的学号: ");
209. scanf("%s", id);
210. clearInputBuffer();
211. Node\* prev = head;
212. Node\* curr = head->next;
213. int found = 0;
214. while (curr != NULL) {
215. if (strcmp(curr->data.id, id) == 0) {
216. prev->next = curr->next;
217. free(curr);
218. printf("学号为 %s 的学生信息已删除。\n", id);
219. found = 1;
220. break;
221. }
222. prev = curr;
223. curr = curr->next;
224. }
225. if (!found) {
226. printf("未找到学号为 %s 的学生。\n", id);
227. }
228. }
229. void updateMenu(Node\* head) {
230. int choice;
231. do {
232. printf("\n--- 更新记录模块 ---\n");
233. printf("  1. 修改学生信息\n");
234. printf("  2. 删除学生信息\n");
235. printf("  0. 返回主菜单\n");
236. printf("---------------------\n");
237. printf("请输入您的选择: ");
238. scanf("%d", &choice);
239. clearInputBuffer();
240. switch (choice) {
241. case 1: modifyStudent(head); break;
242. case 2: deleteStudent(head); break;
243. case 0: break;
244. default: printf("无效选择，请重新输入。\n");
245. }
246. } while (choice != 0);
247. }
248. *// 4. 统计记录模块*
249. void statisticsByClass(Node\* head) {
250. if (head->next == NULL) {
251. printf("当前无任何学生信息。\n");
252. return;
253. }
254. char className[50];
255. printf("请输入要统计的班级名称: ");
256. scanf("%s", className);
257. clearInputBuffer();
258. Node\* p = head->next;
259. int count = 0;
260. while (p != NULL) {
261. if (strcmp(p->data.className, className) == 0) {
262. count++;
263. }
264. p = p->next;
265. }
266. printf("班级 %s 共有 %d 名学生。\n", className, count);
267. }
268. void statisticsByGender(Node\* head) {
269. if (head->next == NULL) {
270. printf("当前无任何学生信息。\n");
271. return;
272. }
273. int male\_count = 0;
274. int female\_count = 0;
275. Node\* p = head->next;
276. while (p != NULL) {
277. if (strcmp(p->data.gender, "男") == 0) {
278. male\_count++;
279. }
280. else if (strcmp(p->data.gender, "女") == 0) {
281. female\_count++;
282. }
283. p = p->next;
284. }
285. printf("学生总数中，男生有 %d 人，女生有 %d 人。\n", male\_count, female\_count);
286. }
287. void statisticsMenu(Node\* head) {
288. int choice;
289. do {
290. printf("\n--- 统计记录模块 ---\n");
291. printf("  1. 按班级统计\n");
292. printf("  2. 按性别统计\n");
293. printf("  0. 返回主菜单\n");
294. printf("---------------------\n");
295. printf("请输入您的选择: ");
296. scanf("%d", &choice);
297. clearInputBuffer();
298. switch (choice) {
299. case 1: statisticsByClass(head); break;
300. case 2: statisticsByGender(head); break;
301. case 0: break;
302. default: printf("无效选择，请重新输入。\n");
303. }
304. } while (choice != 0);
305. }
306. *// 5. 输出记录模块*
307. void displayAllStudents(Node\* head) {
308. if (head->next == NULL) {
309. printf("当前无任何学生信息。\n");
310. return;
311. }
312. printf("\n======================= 所有学生信息 =======================\n");
313. printf("%-15s %-15s %-10s %-10s %-15s\n", "学号", "姓名", "性别", "年龄", "班级");
314. printf("------------------------------------------------------------\n");
315. Node\* p = head->next;
316. while (p != NULL) {
317. printf("%-15s %-15s %-10s %-10d %-15s\n",
318. p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);
319. p = p->next;
320. }
321. printf("============================================================\n");
322. }
323. *// 6. 文件操作模块*
324. void saveToFile(Node\* head, const char\* filename) {
325. FILE\* fp = fopen(filename, "w");
326. if (fp == NULL) {
327. printf("打开文件 %s 失败！\n", filename);
328. return;
329. }
330. Node\* p = head->next;
331. while (p != NULL) {
332. fprintf(fp, "%s %s %s %d %s\n",
333. p->data.id, p->data.name, p->data.gender, p->data.age, p->data.className);
334. p = p->next;
335. }
336. fclose(fp);
337. printf("数据已成功保存到 %s 文件中。\n", filename);
338. }
339. Node\* loadFromFile(const char\* filename) {
340. *// 创建一个带头节点的空链表*
341. Node\* head = (Node\*)malloc(sizeof(Node));
342. if (head == NULL) {
343. printf("内存分配失败！\n");
344. exit(1);
345. }
346. head->next = NULL;
347. FILE\* fp = fopen(filename, "r");
348. if (fp == NULL) {
349. printf("未找到数据文件 %s，将创建一个新系统。\n", filename);
350. return head;
351. }
352. Student temp;
353. while (fscanf(fp, "%s %s %s %d %s", temp.id, temp.name, temp.gender, &temp.age, temp.className) != EOF) {
354. Node\* newNode = (Node\*)malloc(sizeof(Node));
355. if (newNode == NULL) {
356. printf("内存分配失败！\n");
357. continue;
358. }
359. newNode->data = temp;
360. newNode->next = NULL;
361. *// 插入到链表尾部*
362. Node\* p = head;
363. while (p->next != NULL) {
364. p = p->next;
365. }
366. p->next = newNode;
367. }
368. fclose(fp);
369. printf("已从 %s 文件加载数据。\n", filename);
370. return head;
371. }
372. *// 辅助函数：释放链表*
373. void freeList(Node\* head) {
374. Node\* p = head;
375. Node\* temp;
376. while (p != NULL) {
377. temp = p;
378. p = p->next;
379. free(temp);
380. }
381. }
382. main.cpp (主函数文件)
383. #define \_CRT\_SECURE\_NO\_WARNINGS
384. #include "student.h"
385. #define FILENAME "student.txt"
386. void showMenu() {
387. printf("\n========= 学生信息管理系统 =========\n");
388. printf("||                                ||\n");
389. printf("||    1. 录入学生信息             ||\n");
390. printf("||    2. 查询学生信息             ||\n");
391. printf("||    3. 更新学生信息             ||\n");
392. printf("||    4. 统计学生信息             ||\n");
393. printf("||    5. 显示所有信息             ||\n");
394. printf("||                                ||\n");
395. printf("||    0. 退出系统 (Exit)          ||\n");
396. printf("||                                ||\n");
397. printf("====================================\n");
398. printf("请输入您的选择 [0-5]: ");
399. }
400. int main() {
401. *// 程序启动时，从文件加载数据*
402. Node\* head = loadFromFile(FILENAME);
403. int choice;
404. do {
405. showMenu();
406. scanf("%d", &choice);
407. clearInputBuffer(); *// 吸收多余的换行符*
408. switch (choice) {
409. case 1:
410. addStudent(head);
411. saveToFile(head, FILENAME);
412. break;
413. case 2:
414. queryMenu(head);
415. break;
416. case 3:
417. updateMenu(head);
418. saveToFile(head, FILENAME);
419. break;
420. case 4:
421. statisticsMenu(head);
422. break;
423. case 5:
424. displayAllStudents(head);
425. break;
426. case 0:
427. {   FILE\* fp = fopen(FILENAME, "w"); }*// 以写入模式打开文件，会直接清空内容*
428. printf("正在退出系统...感谢使用！\n");
429. break;
430. default:
431. printf("无效的输入，请输入 0-5 之间的数字。\n");
432. break;
433. }
434. printf("\n按 Enter 键继续...");
435. getchar(); *// 暂停程序，等待用户按键*
436. system("cls"); *// 清屏*
437. } while (choice != 0);
438. *// 退出程序前，释放链表内存*
439. freeList(head);
440. return 0;
441. }